日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９９年１２月　１日

出願番号
Application Number: 平成１１年特許願第３４２６５９号

出願人
Applicant (s): インターナショナル・ビジネス・マシーンズ・コーポレイション

２０００年　３月１７日

特許庁長官
Commissioner,
Patent Office 近藤隆彦

出証番号　出証特２０００－３００２５３５

【書類名】　特許願

【整理番号】　JA999703

【特記事項】　特許法第３６条の２第１項の規定による特許出願

【あて先】　特許庁長官　殿

【国際特許分類】　G06F　9/44

【発明者】

【住所又は居所】　インド国５６００７１、バンガロール、クロスロード１４、ドムルユア　レイアウト　オフィス　６４０

【氏名】　ラジェンドラ・クマール・ベラ

【特許出願人】

【識別番号】　390009531

【住所又は居所】　アメリカ合衆国１０５０４、ニューヨーク州アーモンク（番地なし）

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレイション

【代理人】

【識別番号】　100086243

【弁理士】

【氏名又は名称】　坂口　博

【連絡先】　０４６－２７３－３３１８、３３２５、３４３１

【選任した代理人】

【識別番号】　100091568

【弁理士】

【氏名又は名称】　市位　嘉宏

【手数料の表示】

【予納台帳番号】　024154

【納付金額】　35,000円

【提出物件の目録】

【物件名】　外国語明細書　1

【物件名】　外国語図面　1

【物件名】　外国語要約書　1

【包括委任状番号】　9304391

【包括委任状番号】　9304392

【プルーフの要否】　要

【書類名】　外国語明細書

1. Title of Invention

METHOD FOR DETERMINING THE SYNTACTIC CORRECTNESS OF EXPRESSIONS

2. Claims

(1) A method for determining, in a computer environment, the syntactic correctness of expressions, such as may be used in computer programs, said method comprising the steps of:

(a) defining delimiter character set;

(b) defining two string arrays, called FromStrg and ToStrg;

(c) creating a string, for convenience here, called strg from the given expression in accordance with a set of predetermined rules; and

(d) reducing said string strg in accordance with a set of predetermined simplifying rules where elements of FromStrg, if found in strg, are replaced with corresponding elements from ToStrg.

(2) A method as claimed in claim 1 and applied to examples which may, by analogy, be treated in analogous fashion to an expression, such as filenames, directory names, variable names, etc.

(3) An apparatus adapted to determining, in a computer environment, the syntactic correctness of expressions, such as may be used in computer programs, said method comprising the steps of:

(a) defining delimiter character set;

(b) defining two string arrays, for convenience here, called FromStrg and ToStrg; (c) creating a string, for convenience here, called strg from the given expression in accordance with a set of predetermined rules; and (d) reducing said string strg in accordance with a set of predetermined simplifying rules where elements of FromStrg, if found in strg, are replaced with corresponding elements from ToStrg.

(4) A computer program product for determining, in a computer environment, the syntactic correctness of expressions, such as may be used in computer programs, said method comprising the steps of:

(a) defining delimiter character set;

(b) defining two string arrays, called FromStrg and ToStrg;

(c) creating a string, for convenience here, called strg from the given expression in accordance with a set of predetermined rules; and

(d) reducing said string strg in accordance with a set of predetermined simplifying rules where elements of FromStrg, if found in strg, are replaced with corresponding elements from

ToStrg.

3. Detailed Description of Invention

Technical Field of the Invention

The present invention relates to the determination of the syntactic correctness of expressions, such as may be used in computer programs.

Background Art

Algebraic expressions are typically used in computer programs to assign values to variables. These expressions normally occur on the right hand side of an assignment statement, and in particular after an assignment operator. The assignment operator most often used is an "=" sign. They also occur in parameter lists of functions, in conditional statements, etc., in computer programs. An example of an assignment statement is:

S = a + b*c + fn(a, b+c, d) + func(a, fn(c,d,e)) (1)

wherein variable S is assigned a value, the value being dependent on a, b, c and d, each of which is either a number or a variable, and fn() and func(), which are functions.

Prior methods of determining the syntactic correctness of expressions routinely break up the expression into tokens and create a tree, where the tokens sit at the nodes of the tree.

A number of rules of how to deal with those tokens, that have been created, are then applied.

Disclosure of the Invention

It is an object of the present invention to provide an alternate and/or improved method of determination of the syntactic correctness of expressions.

The present invention deals with the entire expression in its operations. It essentially mimics the way humans visually interpret and check an expression's syntax. Thus the invention looks for substrings (character sequences), which are legitimate and replaces them with a shorter but semantically equivalent substring. It also looks for character sequences which are illegitimate, and replaces them with the substring "?" to indicate that it has met with an illegal sequence. It does these operations repeatedly, and in predetermined sequences, till no further changes in the string can be made; that is, the string achieves a constant length.

Detailed Description including Best Mode

Referring to Fig. 1, a flow diagram is shown of a method 100 of determining whether an algebraic expression is syntactically correct.

Starting at step 10, a set of characters is defined called a delimiter character set. The delimiter character set is compactly

expressed as a string, where each character within the quotation marks is a delimiter character. An example of a delimiter character set for an algebraic expression is "+-*/,()". It can additionally include the modulo operator "%".

Step 20 defines a FromStrg[] string array and a ToStrg[] string array. With respect to the delimiter character set defined, the two string arrays are defined as (in the notations of C programming language):

```
FromStrg[] = {")(", "(v)()", "(v)(v)", "v(v)", "v(#)", "v()",
"(v)", "vv", "v[v]", "(v,v)", "(+v)", "(-v)",
"v*v", "v/v", "v+v", "v-v", ",v,", ""};
ToStrg[]
= {"?", "?", "?", "v", "v", "v", "v", "v", "v", "(#)", "(v)",
"(v)",
"v", "v", "v", "v", ",", ""};
```

There is a one-to-one correspondence between the elements of the two arrays,
FromStrg[] and ToStrg[], such that the i-th element in the FromStrg[] array, corresponds to the i-th element in the ToStrg[] array. The number of elements in the FromStrg[] array are obviously equal to the number of elements in the ToStrg[] array. The NULL string at the end of each array FromStrg[] and ToStrg[], is used to indicate the end of the array.
In step 30, a new character string strg is created from the algebraic expression being checked by performing the following substeps:

In substep 31 (not shown), it is determined whether the given algebraic expression begins with a unary + or - operator. If this is so, then the expression is prefixed with the numeral 0. Alternatively, the algebraic expression may be enclosed in brackets. Furthermore, all blank (or space) characters are removed from the algebraic expression. A counter variable i is initialized to 0.

In substep 32 (not shown), the expression is scanned from left to right, character by character, until a delimiter character is found or the end of the expression is reached.
A variable delimiter character (is set equal to the delimiter character found.

(a) If no characters are found before the delimiter character (, then the i-th character in the character string strg is set equal to the delimiter character as follows: strg[i] = (. The counter variable i is also incremented by 1 and the procedure continues to substep 33 (not shown).

(b) If one or more characters are found before the delimiter then it is determined whether it/they form(s) a valid variable name, function name or number. If the character(s) is/are valid, then a character "v" is put into string position strg[i]. Alternatively a character "?' is put into string position strg[i]. The following string position in the string strg[] is filled with the delimiter character ??as follows: strg[i+1] = ?. The counter variable i is increased by 2 and the procedure continues to substep 33.

Substep 33 determines whether the end of the expression has been reached. If this is not so, then the procedure continues to substep 32, assuming that the expression now begins at the character just following the delimiter character ?.
As an example, the algabraic expression obtained from the assignment statement (1) is as follows:

a + b*c + fn(a, b+c, d) + func(a, fn(c,d,e)) (2)

and the following string strg[]will be produced by following substeps 31 to 33:

v+v*v+v(v,v+v,v)+v(v,v(v,v,v)) (3)

The method 100 continues to step 40 where it is determined whether the constructed string strg[] contains one or more "?" characters. If the string strg[]contains at least one "?" character, then the algebraic expression is syntactically incorrect, and the method 100 ends in step 90 by returning that the expression is not correct. Alternatively, the method 100 continues to step 50. Step 50 initializes a variable size as the length of the of the string strg, including an end of string character. Step 51 sets the counter variable i equal to -1, while step 52 increments the counter variable i by 1.

Step 53 determines whether the counter variable i is still smaller than n, where n is the dimension of the FromStrg[ ] and ToStrg[] arrays. If this is affirmative, then step 54 replaces each

occurrence of the string FromStrg[i] in string strg with ToStrg[i]. Steps 52 to 54 are repeated until step 53 determines that the end of the FromStrg[i] array has been reached. The method 100 then continues to steps 55 and 56 where it is determined whether the resulting string strg is the same size as before steps 51 to 54, by comparing it with the variable size. If the length of the string strg has changed, then the method 100 returns to step 51 after the variable size has been set to the new length of the string strg. If the length of the string strg has not changed, it means that the string strg has been reduced as far as possible and the method 100 continues to step 58 where it is determined whether the string strg equals the character "v". Only if strg="v" is the algebraic expression syntactically correct and the method 100 ends in step 80. (The significance of this is that any syntactically correct expression must eventually be reducible to a number or a variable.) Alternatively the algebraic expression is syntactically incorrect, and the method 100 ends in step 90 by returning that the expression is not correct.

Examples of algebraic expressions that are not syntactically correct are:

a*+b : there are two operators between variables a and b;

*a*b : an expression can not start with a multiplication operator;

a(b+c) : there is no operator between the variable a and the operator (;

a*(b+c)) : the last) operator does not have a matching (
operator.

Source or psuedo-code for performing steps 50 to 90 of the method 100 is as follows:

```
// strg[] is the character array derived from the given expression
// using steps 1-3 described above.
// size is the size of the string strg.
// n is the dimension of FromStrg[] and ToStrg[] arrays.
// ChangeSubstrg() replaces FromStrg[i], if found in strg with
// ToStrg[i].
// cond is a boolean flag which saves the result of the iteration.
// It is TRUE if the strg is syntactically correct, else FALSE.

size = strlen(strg) + 1;
cond = TRUE;
while (cond) {
  i = -1
  while (++i < n) {
   while (strstr(strg, FromStrg[i]))
   ChangeSubstrg(strg, FromStrg[i], ToStrg[i];
  }
 j = strlen(strg) + 1;
 if (size == j) break;
 size = j;
}
if (strcmp(strg, "v") != 0) cond = FALSE;
```

The method 100 may be viewed as adding to the list of isxxx() functions in modern compiler libraries, such as (in C), isalpha(), isdigit(), etc. The present method 100 may be encoded into an isexpression() function.

In an alternative embodiment, the method 100 is used to determine the syntactic correctness of a file name/directory expression. Say, a filename expression can have one of the following three syntaxes:

(i) <drive name">:/<dir name>/<subdir name>/.../<subdirname>/<filename>/.<ext">

(ii) <filename>.<ext>

(iii) <filename>

Following the method 100, and in particular step 10, a delimiter character set is defined as "./". Step 20 defines the following string arrays:

FromStrg[] = {"f.f.f", "f.f/", "f.f", "/f/", "f:", "/#", "#/f", ""}

and

ToStrg[] = {"?", "?", "f", "/", "#", "?","f", ""}

Here f represents a (sub)directory name, filename or a filename

extension. Hence if filename is syntactically correct, it will initially produce the string

"f:/f/f/.../f/f.f", "f.f" or "f" respectively, for the syntaxes (i), (ii), (iii)
noted above.

Only if the filename is syntactically correct, then a string "f" will result from the method 100. Step 58 determines whether the string strg has been reduced to the
character "f" (in Fig.1 it means replacing "v" by "f").

This embodiment may be coded as an IsFilename() function, which takes a character string, presumed to be a filename, as its input.

In yet another embodiment, the method 100 is used to determine whether an expression forms a valid variable or function name. It is assumed that this expression must start with an alphabetical character or an underscore, followed by a sequence of characters, each of which can be an alpha-numeric character or an underscore. In step 30, scanning the expression string from left to right character by character, if the character is a number, it is replaced with the character "n". Similarly, if the character is an alphabet character or an underscore, it is replaces with the character "a". If the character is anything else, then it is replaced with the character "?".

In this embodiment, the delimiter character set defined is empty, while the following two string arrays are defined in step 20:

FromStrg[] = {"aa", "an", ""}

and

ToStrg[] = {"a", "a", ""}

Only if the string expression is a variable name will step 58 determine that the string strg has reduced to the character "a" (in Fig.1 it means replacing "v" by "a") and proceed to step 80 for correct expressions.

This embodiment can be coded as an IsVar() function, which takes the character string, presumed to be a variable, as its input. Embodiments of the invention can be implemented within compilers, for example. As is well known, a compiler generates machine executable object code from high-level source code, written in languages such as C++ and Java™.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

4. Brief Description of Drawings

Fig. 1 is a flow diagram of a method of determining whether an algebraic expression is syntactically correct.

【書類名】　外国語図面

【図１】

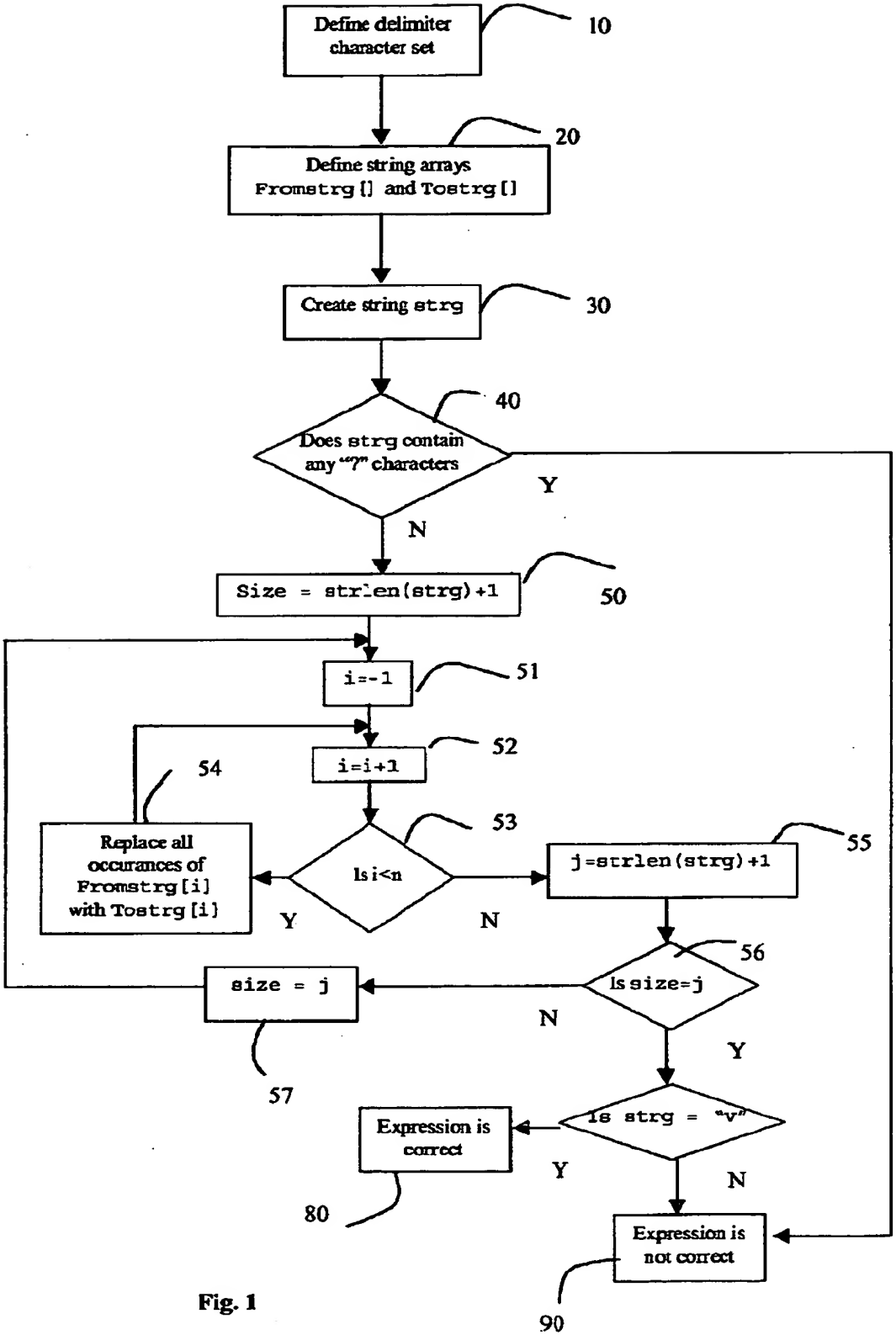
Define delimiter character set
10
Define string arrays Fromstrg[] and Tostrg[]
20
Create string strg
30
Does strg contain any "?" characters
40
Y
N
Size = strlen(strg)+1
50
i=-1
51
i=i+1
52
54
Replace all occurances of Fromstrg[i] with Tostrg[i]
Is i<n
53
Y
N
j=strlen(strg)+1
55
Is size=j
56
N
Y
size = j
57
Is strg = "v"
Expression is correct
Y
N
80
Expression is not correct
90

Fig. 1

【書類名】　外国語要約書

1. Abstract

It is an object of the present invention to provide an alternate and/or improved method of determination of the syntactic correctness of expressions.

The present invention deals with the entire expression in its operations. It essentially mimics the way humans visually interpret and check an expression's syntax. Thus the invention looks for substrings (character sequences), which are legitimate and replaces them with a shorter but semantically equivalent substring.
It also looks for character sequences which are illegitimate, and replaces them with the substring "?" to indicate that it has met with an illegal sequence. It does these operations repeatedly, and in predetermined sequences, till no further changes in the string can be made; that is, the string achieves a constant length.

2. Representative Drawings

Fig. 1

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　１９９０年１０月２４日
［変更理由］　新規登録
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレイション